CST1
COMPUTER SCIENCE TRIPOS Part IB

Monday 5 June 2023 13:30 to 16:30

COMPUTER SCIENCE Paper 4

*Answer **five** questions.*

*Submit the answers in five **separate** bundles, each with its own cover sheet. On each cover sheet, write the numbers of **all** attempted questions, and circle the number of the question attached.*

You may not start to read the questions printed on the subsequent pages of this question paper until instructed that you may do so by the Invigilator

STATIONERY REQUIREMENTS
Script paper
Blue cover sheets
Tags

SPECIAL REQUIREMENTS
Approved calculator permitted

1 Compiler Construction

(*a*) Describe the inputs and outputs of a lexer and parser. [2 marks]

(*b*) Here are three grammars for languages with branching, sequencing and variables:

Grammar 1:
```
E -> if E then E else E
E -> E then E end
E -> id
```

Grammar 2:
```
E -> if E then E else E end
E -> E then E end
E -> id
```

Grammar 3:
```
E -> if E then E else E
E -> do E then E
E -> id
```

(*i*) For each grammar, state whether it is ambiguous, giving an example if it is. [3 marks]

(*ii*) For each grammar, state whether it is in LL(1), giving a reason if it is not. [3 marks]

(*c*) Explain the roles of the ACTION and GOTO tables in the LR parsing algorithm, describing the entries and indexes for each table. [4 marks]

(*d*) Compilers sometimes simplify expressions to make type checking easier or to generate more efficient code. Here are two potential Slang simplifications:

expression	simplifies to	expression	
`if true then` $\texttt{e}_1$ `else` $\texttt{e}_2$	$\rightarrow$	$\texttt{e}_1$	
`(fun (x:t) ->` $\texttt{e}_1$`)` $\texttt{e}_2$	$\rightarrow$	$\{\texttt{e}_2/\texttt{x}\}\texttt{e}_1$	(substitution)

A simplification $e_1 \rightarrow e_2$ is *correct for type checking* if e_1 and e_2 have the same type (or are both ill-typed), and *correct for optimization* if e_1 and e_2 have equivalent behaviour.

(*i*) For each simplification, explain under what circumstances it is correct for type checking. [4 marks]

(*ii*) For each simplification, explain under what circumstances it is correct for optimization. [4 marks]

2 Compiler Construction

Here is an OCaml definition of the Ackermann function, `ack`:

```
let rec ack m n =
   if m = 0 then n+1
   else if n = 0 then ack (m-1) 1
   else ack (m-1) (ack m (n-1))
```

(*a*) You would like to run `ack` on an old system with limited stack space and no support for closures, and decide to rewrite it in stages.

(*i*) Rewrite the `ack` function to produce a function `ack_cps` in continuation-passing style so that the function

```
let ack_1 m n = ack_cps m n (fun x -> x)
```

produces the same results as the function `ack`. [5 marks]

(*ii*) Eliminate higher-order functions from your answer to Part (*a*)(*i*) by rewriting `ack_cps` as a function `ack_cps_dfn` in *defunctionalized* form so that the function

```
let ack_2 m n = ack_cps_dfn m n ID
```

produces the same results as the function `ack`. [5 marks]

(*iii*) Convert your answer from Part (*a*)(*ii*) to a function `ack_cps_dfn_list` that uses standard lists rather than custom data types [5 marks]

(*b*) Briefly comment on the way that `ack` and the transformed implementations in Parts (*a*)(*i*), (*a*)(*ii*) and (*a*)(*iii*) use memory, making reference to the stack and heap. [5 marks]

3 Concepts in Programming Languages

(*a*) A programmer reviewing a multi-file Java program containing

```
class A { protected final int a; public A() { a=3; }
          public int query() { return a; }}
class B extends A { protected final int b;
                    public B() { b=1; }
                    public int query() { return a*b; }}
```

reasons that `int q(A x) { return x.query(); }` "must always return 3".

(*i*) Giving reasons, agree or disagree with the programmer. [3 marks]

(*ii*) Might appropriately inserting `final` change your answer? [2 marks]

(*iii*) Give a name for the property being discussed, along with a brief statement of what it requires. [3 marks]

(*b*) A common bug-pattern can be exemplified in pseudo-Java by

```
class Database {
   private List<MyRecord> data;
   public void insert(MyRecord x) { data.add(x); }
   public void tidy() { /* may mutate items in data */ }
}
class User {
   public void foo(Database db) {
      var x = new MyRecord(...);
      int memo = x.field;
      db.insert(x); ...; db.tidy();
      assert (x.field == memo) raising "SomeOneIsConfused";
   }
}
```

(*i*) Why might the programmer have added the assertion? [2 marks]

(*ii*) Give two possible ways a "coding standards" document for the project could have protected against the assertion failing, in both cases summarising any modifications required to the above code. [6 marks]

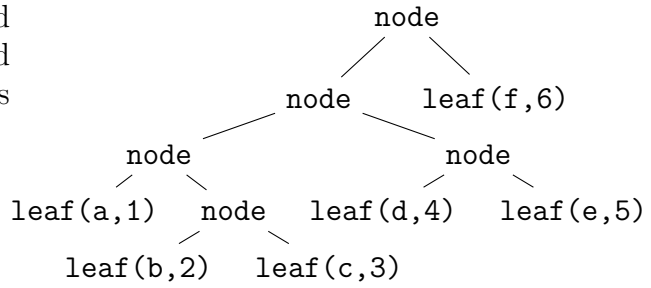
(*iii*) Name a language which provides a compile-time mechanism to express such coding-standard requirements. Indicate briefly, and in no particular syntax, the coding concept involved. [4 marks]

4 Prolog

In your answers ensure each relation has a comment giving a declarative reading of its behaviour. Avoid unnecessary use of *cut* and do not use extra-logical relations such as `findall`, `assertz` and `not` (`\+`). Built-in library relations should not be assumed. The Prolog operator in `A\=B`, meaning `A` will not unify with `B`, may be used.

(*a*) Explain Prolog's process of *unification*. What situation would an *occurs check* guard against? [3 marks]

(*b*) Assume `node(Left,Right)` and `leaf(Name,Value)` compound terms are used to represent trees such as:



Define a relation `lookup(+Tree,?Name,?Value)` which finds the value(s) associated with a given name in trees of the above form. [3 marks]

(*c*) Given a list of atoms, `L1`, define a relation `rle(+L1,?L2)` which run-length *encodes* `L1` into `L2`. For example, `rle([a,a,b,c,a,a,a],L)` should succeed with `L=[2*a, 1*b, 1*c, 3*a]`. Giving reasons, indicate for your answer whether a query `rle(L,[2*a, 1*b, 1*c, 3*a])` would succeed with `L=[a,a,b,c,a,a,a]`. [5 marks]

(*d*) Complementary to `rle/2`, define a relation `rld(+L1,?L2)` which *decodes* a run-length-encoded list `L1` as defined in part (c) into `L2`. [4 marks]

(*e*) The Prolog relations below, given a query `alter_list([2,4,6],L)`, will succeed with `L=[a,a,b]`. Use an additional difference-list argument to accumulate the execution path through the Prolog clauses. Number the clauses 1 to 4 such that `alter_list([2,4,6],L,Path-[])` will succeed with the sequence of clauses as a list of integers in Path, i.e. with `L=[a,a,b]` and `Path=[4,1,4,1,4,2,3]`.

```
change(N,a) :- N < 5.
change(N,b) :- N >= 5.

alter_list([],[]).
alter_list([H1|T1],[H2|T2]) :- change(H1,H2), alter_list(T1,T2).
```

[5 marks]

5 Programming in C and C++

(*a*) 14-bit words are used to represent a certain set of natural numbers including zero. The least-significant five bits contain an unsigned binary-encoded mantissa value. The remaining nine bits represent an unsigned binary-encoded, bitwise left shift to be applied to the mantissa to obtain the represented value.

(*i*) Give two functions, coded in C, that respectively convert an encoded value to its nearest 32-bit unsigned integer and to its nearest double-precision floating-point number. What problem(s) arise? [6 marks]

(*ii*) A packed array of such 14-bit words is stored in memory. Packed means no memory bits are unused, so the stored words may cross byte boundaries. Write a C function to implement the update operation for a 14-bit word held in the array. Its three arguments are an `unsigned char *` pointer to the base of the packed array, an integer index and an integer holding the 14-bit word to be stored. You may assume unaligned loads and stores of 32-bit words is supported. [6 marks]

(*b*) All calls to `malloc()` in a user program in C are to be replaced with calls to `my_malloc`.

(*i*) Provide an implementation of `void *my_malloc(size_t)` that invokes the system's underlying `malloc` but which adds 16 bytes of padding at the start and end of each allocated region which is initialised with a distinctive bit pattern. [3 marks]

(*ii*) Provide a companion `my_free` function that checks for any changes to the starting pattern, reporting appropriately, or else continues to invoke the system's `free`. [3 marks]

(*iii*) What might be the motivation for introducing `my_malloc`? [2 marks]

6 Programming in C and C++

(*a*) Describe the two main forms of dynamic storage in languages such as C and OCaml? How does each potentially interact with a garbage collector? [4 marks]

(*b*) Does any aspect of C or C++ prevent a compiler from implementing the 'tail-recursion optimisation' exploited in functional languages and would this tend to reduce memory leak problems (e.g. a web server running out of memory)? [2 marks]

(*c*) An 'in-core' database system uses a number of client processes that share a segment of memory with a server process which manages data storage. Assuming there are currently four client processes running, describe or compute the total number of user-space segments likely to be allocated in physical memory. How might memory in the shared segment be allocated? [4 marks]

(*d*) C++ allows multiple inheritance, e.g.

```
class D: B,C { int d1, d2; };
```

where classes `B` and `C` have previously been defined. [*Note:* Your answer need not consider access qualifiers nor member functions.]

(*i*) First assume that classes `B` and `C` do not inherit from other classes. Give C code which summarises the storage layout of a variable `x` of type `class D`. [3 marks]

(*ii*) Given `class D *p`, how can C++-style "cast to pointer to base class", exemplified by `(B *)p` and `(C *)p`, be achieved by C code? [2 marks]

(*iii*) Now suppose that classes `B` and `C` both inherit from class `A`. Explain a conceptual choice which arises as to how data members of class `A` appear within the storage layout for a variable `x` of type `class D`. [2 marks]

(*iv*) Explain how the `virtual` keyword allows a programmer to select between the alternatives in Part (*d*)(*iii*). [3 marks]

7 Cybersecurity

The following Python program (`imports` omitted for brevity) takes a user-supplied `fields.txt` input file containing two text fields, one per line. By concatenating them with template fragments (not fully shown for brevity), it writes out a temporary LaTeX file that places those fields in specific positions on the page. It then compiles the LaTeX into a pdf, intended to be overprinted onto an existing form. Assume the user has no direct access to the file system: the user interacts with a web page that writes the user-supplied fields, as supplied, to the `fields.txt` file. [*Note:* In Python, when the end of the file is reached, calling `readline` returns empty string. The default encoding used by `readline` is UTF-8.]

```
1  t_before = "\\documentclass{article} \\begin{document} ..."
2  t_middle = " ... "
3  t_after = " ... \\end{document}"
4  with open("fields.txt", "r") as fields:
5       f0 = fields.readline()
6       f1 = fields.readline()
7  with open("form-content.tex", "w") as latex:
8     latex.write(t_before + f0 + t_middle + f1 + t_after)
9  os.system("pdflatex form-content.tex")
```

(*a*) Is it possible to produce a `fields.txt` input file that will cause a malfunction while executing lines 1–8? If yes, produce such input; otherwise justify why this is not possible. State your assumptions explicitly. [3 marks]

(*b*) Is it possible to produce a `fields.txt` input file that will cause a malfunction while executing line 9? If yes, produce such input; otherwise justify why this is not possible. State your assumptions explicitly. [3 marks]

(*c*) Amend the program so that it won't malfunction on the specific inputs you provided in parts (*a*) or (*b*). [*Note:* Ad-hoc fixes that won't fix the general problem are deemed sufficient for full marks here—but see part (*e*).] [*Hint:* At least one of parts (*a*) or (*b*) admits a "yes" answer.] [3 marks]

(*d*) Is it possible for an attacker to exploit this program to execute an arbitrary command, and under which circumstances? If yes, produce an input file that will execute `/tmp/payload`. If not, justify why this is not possible. State your assumptions explicitly. [3 marks]

(*e*) After the above program caused embarrassment, you are called in to offer security training to the company's programmers. State the two most important points you are going to make. For each, clearly explain the remedial approaches you recommend, discussing them in the context of the above listing. [8 marks]

8 Cybersecurity

(*a*) Compare and contrast, under at least four relevant aspects, the mechanical implementation of a master-keyed lock system against one using electronic key fobs. Then give one scenario where you would recommend the mechanical implementation and one where you would recommend the electronic one, justifying why. [6 marks]

(*b*) A building with 24 door locks, D0–D23, uses a mechanical (pin-tumbler) master key system. Locks have 5 pin stacks, there are 8 possible cut depths per stack from 0 (shallowest) to 7 (deepest) and each stack contains at most one master pin. Pretend, unrealistically, that all cut depths are usable. Each door has its own differ key that opens just that door. Doors D10–D19 may also be opened by master key 36735. All doors, D0–D23, may also be opened by grand master key 12735.

(*i*) In the lock for door D12, what is the minimum and maximum possible number of pin stacks containing a master pin? Explain your reasoning. [2 marks]

(*ii*) Door D13 is opened by differ key 32737. List *all* the other keys that also open that door. Mention and explain your assumptions. [4 marks]

(*iii*) The occupant of the D16 office, who legitimately owns the D16 differ key 32535 but not the master nor the grand master key, wants to open door D07 using the Matt Blaze privilege escalation attack, visiting door D07 only once in order not to arouse suspicion. She processes the pins left to right, processes the pin depths from shallowest to deepest and uses an optimal strategy. Assume there are no master pins in the D16 lock other than those necessary for it to be opened by the indicated differ key, master key and grand master key. List, in order, every key that the attacker must create and try, mentioning which lock she tries it on and whether it opens, and giving a clear running explanation of what she does. How many keys exactly (not an upper bound) does she need to test on D16, and how many on D07, before successfully opening D07? [8 marks]

END OF PAPER